Java Stored Procedures - Q & A

Josip Pojatina
mStart d.o.o. (Agrokor ICT)
josip.pojatina@mStart.hr

Table of contents

About mStart

- *Agrokor ICT 1.7.2010. changed the name into mStart d.o.o.*
- *Operates as an independent company within the Agrokor Group*
- *Main goal is to provide support for 200+ clients*

About the author

- *Architecture/design/optimization/development/administration*
- *15+ years with Oracle RDBMS*
- *10+ years of experience with optimization of the large sites based on the Oracle technology (Oracle Db, Web Logic, Oracle Service Bus (OSB), Java/JRockit JVM)*
- *Red Hat / Oracle Linux, IBM AIX*
- *speciality - Oracle CBO, PL/SQL i Java store procedures*
- *Oracle Retail*
- *Oracle eBS*

Introduction

- *What will be covered in this presentation:*
  - *How to load Java stored procedures in Oracle db*
  - *Where to put Java stored procedures in db*
  - *How to debug Java stored procedures*
  - *How to profile Java stored procedures*
  - *When to use Java stored procedures*

Introduction

- *Introduced with Oracle 8i rdbms back in 1999*
- *Java in Db is running on top heavily customized JVM*
- *Lag between current standard Java version (Java 8) and Db version (Java 6 in 12c, possible upgrade to Java 7, Java 5 in 11.2.0.3, possible upgrade to Java 6 in 11.2.0.4)*
- *Tightly integrated with Oracle rdbms*
- *Specific architecture as consequence of tight integration*
- *Requires knowledge of Oracle (DBA) and Java*
- *Not widely used as should be due to the required skills*

Introduction

- *Code is running on all platforms where Oracle Db is running*
- *Very robust and scalable*
- *Sessined based architecture*
- *Harder to debug than java outside the Db (besides Java knowledge, require DBA skills)*
- *Java code intergrated with PL/SQL and SQL*
- *Lack of threading support (nonpreemptive scheduler)*
- *GUI materialization not possible*
- *JVM is running in SGA*

Introduction

- *Despite session model, only statics and private states are in session space (everything else is shared)*
- *Robustness as result that each session has it's own JVM*
- *Security model in accordance with Oracle Db security*
- *JVM upgrade possible only as part of Db upgrade*
- *Three method of execution Java code*
  - *interpreted*
  - *JIT (from 11.1g)*
  - *native*

Introduction

- *Many ways to load Java in Db*
  - *DDL (create java, alter java)*
  - *loadjava utility*
  - *JDeveloper (uses loadjava behind the scene)*
  - *dbms_java.load_java*
- *Three ways to invoke Java in the Db*
  - *PL/SQL wrapper approach (most common, problem with call specs)*
  - *OJVMJAVA command line utility*
  - *Client-side approach (JPublisher)*

JVM version in Oracle Db

- *What Version of Java is Compatible With The Database JVM (Doc ID 438294.1)*
  - *DB Version 9.2 - Java 1.3.1*
  - *DB Version 10.2 - Java 1.4.2*
  - *DB Version 11.1 - Java 1.5.0*
  - *DB Version 11.2 - Java 1.5.0 (1.6 from PS 11.2.0.4)*
  - *DB Version 12.1 - Java 1.6 or 1.7 (Whichever version is enabled)*
- *How To Determine The JDK Version Used by the Oracle JVM in the Database (Doc ID 131872.1)*

JVM version in Oracle Db

```
create or replace and compile java source named "GetVersion" as
        public class GetVersion
        {
          public static void printVersion()
          {
            String ver;
            ver = System.getProperty("java.version");
            System.out.println("The JDK version is " + ver);
          }
        }
        /
 PLSQL Wrapper
create or replace procedure Get_JDK_Version is language java
        name 'GetVersion.printVersion()';

SQL>   set serveroutput on
SQL>   call dbms_java.set_output(1000);
SQL>  execute Get_JDK_Version;
The JDK version is 1.6.0
```

Debug - how to

- With classic way of debugging in Oracle we won't have success as Oracle debugger cannot see inside the JVM (JVM is black box for Oracle debugger).
- That can be seen in the following examples:
  - dbms_debug
  - hierarchical profiler
  - 10046 trace event

Debug - dbms_profiler

UNIT_TYPE	UNIT_OWNER	UNIT_NAME	LINE#	TOTAL_OCCUR	TOTAL_TIME
ANONYMOUS BLOCK	<anonymous>	<anonymous>	1	0	0
ANONYMOUS BLOCK	<anonymous>	<anonymous>	3	0	0
ANONYMOUS BLOCK	<anonymous>	<anonymous>	4	0	641
ANONYMOUS BLOCK	<anonymous>	<anonymous>	5	2	101087
ANONYMOUS BLOCK	<anonymous>	<anonymous>	6	1	4938
ANONYMOUS BLOCK	<anonymous>	<anonymous>	7	0	0
PROCEDURE	SCOTT	TESTSPEED9	1	1	12605740852

Debug - hprofiler

- 1 __anonymous_block
- 2 __plsql_vm
- 3 SCOTT TESTSPEED9 PROCEDURE TESTSPEED9
- 4 sys dbms_hprof package body start_profiling
- 5 SYS DBMS_HPROF PACKAGE BODY STOP_PROFILING

Debug - 10046 trace file

- Elapsed times include waiting on following events:

Event waited on	Times Waited	Max. Wait	Total Waited
SQL*Net message to client	2	0.00	0.00
SQL*Net message from client	2	6.15	11.94
row cache lock	4	0.00	0.00
DFS lock handle	2	0.00	0.00
OJVM: Generic	10	1.00	9.99

Debug - two approaches

- From Oracle neutral Java IDE (like Netbeans, Eclipse, JetBrains ...)
  - plus:
    - easy to setup (just add the code fragment from the next slide inside Java Stored Procedure)
  - minus:
    - exceptions in Java Stored Procedures are not properly returned.
    - Creating table for java error table as workaround

Debug - two approaches

```
if (System.getProperty("oracle.jserver.version") != null)
{
/*
 * Inside DB, already connected---> use default connection
 */
conn = DriverManager.getConnection("jdbc:default:connection:");
}
else
{
/*
 * Code is running out of DB. You have to connedt first
 */

 DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
 conn =
        DriverManager.getConnection("jdbc:oracle:thin:@orcl.hroug.hr:1521/hroug", "scott",
          "xxxxx");
}
```

Debug - two approaches

- From Oracle JDeveloper
  - plus:
    - Provides the same feeling as debug client side Java apps (all info provided including Oracle types, collections...)
  - minus:
    - Not easy to setup
    - Close work with DBA required

Where to implement Java st. proc.

- As a Java method with PL/SQL wrapper
  - the most common approach
  - Java methods callable from SQL, PL/SQL procedures/functions/package/trigger
- As a user object type
  - member functions

Where to implement Java st. proc.

```
create or replace procedure testspeed2 as
language java name 'TestSpeed.main(java.lang.String[])';

public class TestSpeed {

    public static void main(String args[]){
        float x;
        for(int i = 1; i <= 10000000; i++ ){
          x = i / 1000;
        }
    }
}
```

Where to implement Java st. proc.

```
create or replace type testspeed as object
(
who varchar2 (100),
member function get_speed
return varchar2 as language java
name 'TestSpeed.getSpeed()
    return java.lang.String'
    ...
)
    instantiable final;
```

Cases for Java Stored Procedures

- *Calling EJB from Oracle JVM*
  - *cooperation between the database and the middle tier to provide business service*
- *HTTP Call-out*
  - *Java apps running inside the Db can invoke Web components such as JSP/Servlets etc. running in the Middle Tier by using HTTP/HTTPS*
  - *notifications*

Cases for Java Stored Procedures

- *JMS in the Database*
  - *JMS can run against different messaging systems*
  - *unlike Oracle AQ/Streams, widely used as a standard way to excehange messages against different systems*
  - *uses AQ infrastructure in the Db*

- *Calling Non-Oracle Db through the JDBC*
  - *by importing appropriate jdbc driver for target DB*
  - *simpler solution and faster solution then Heterogenous Services*

Cases for Java Stored Procedures

- *RMI (Remote Method Invocation)*
  - *accessing external proprietary systems*
  - *acomplish different actions on the remote server*
- *Calculations/sorting/number crunching*
  - *Native, ahead of time compiled Java*
- *Non Java Languages in the Db*
  - *JVM runnable languages like Python, Groovy etc.*

Cases for Java Stored Procedures

- *Non Java Languages in the Db*
  - *JVM runnable languages like Python, Groovy etc.*
- *XML processing*
  - *especially interesting from 11g+ Db version due to the new data types and the new engine for XML*

Cases for Java Stored Procedures

- *Extending existing functionalities*
  - *utl_file*
  - *dbms_mail*
- *Getting enterprise features on Standard edition of Oracle Db*
  - *data encryption*
  - *network encription*
  - *replication*
  - *huge savings in licenses, but need to maintance added functionalities*

Performance tests

- Test Java code:
- public class TestSpeed {
- public static void main(String args[]){
- float x;
- for(int i = 1; i <= 10000000; i++){
- x = i / 1000;
- }
- }
- }

Performance tests - CPU intensive

- *Java stored procedure: interpreted*

- *JPOJATINA@test> exec testspeed2;*
- *PL/SQL procedure successfully completed.*
- *Elapsed: 00:00:01.87*

- *Nakon JIT-a:*
- *Elapsed: 00:00:00.04*

Performance tests - CPU intensive

- *Java stored procedure: native compile (ahead of time compilation)*

- *JPOJATINA@test> exec testspeed2;*
- *PL/SQL procedure successfully completed.*
- *Elapsed: 00:00:00.03*

Performance tests - CPU intensive

- C code compiled on IBM AIX (XLC Compiler)
- #include <stdio.h>
- void main(void)
- {
- double x;
- for (int i = 1; i <= 10000000; i++)
- {
- x = i / 1000;
- } }

Performance tests - CPU intensive

- C code compiled on IBM AIX (XLC Compiler)
- oracle@xxx-xxx-xx-1p:/tmp > time speedtest

- real 0m0.076s
- user 0m0.043s
- sys 0m0.001s

Performance tests - CPU intensive

- PL/SQL code

```
CREATE OR REPLACE PROCEDURE test_speed AS
v_number NUMBER;
begin
FOR i IN 1 .. 10000000 LOOP
v_number := i / 1000;
end loop;
end;
```

Performance tests - CPU intensive

- *PL/SQL code - interpreted*

- *exec test_speed;*
- *1.768 sec*

- *PL/SQL code - native level 3*

- *exec test_speed;*
- *elapsed: 00:00:01.73*

Performance tests - CPU intensive

- create or replace procedure test_speed8 as
- v_number simple_double :=0;
- x simple_double := 1000;
- y simple_double := 0;
- begin
- for i in 1 .. 10000000 loop
- y := i;
- v_number := y / x;
- end loop;
- end;

Performance tests - CPU intensive

- *PL/SQL code optimized - interpreted*

- *exec test_speed8;*
- *PL/SQL procedure successfully completed.*
- *elapsed: 00:00:00.65 - Elapsed: 00:00:00.81*
-

- *PL/SQL code optimized - native compile*

- *Elapsed: 00:00:00.88*

Performance tests - CPU intensive

- *Java - client side code*

- *time java TestSpeed*

- *real 0m0.102s*
- *user 0m0.090s*
- *sys 0m0.019s*

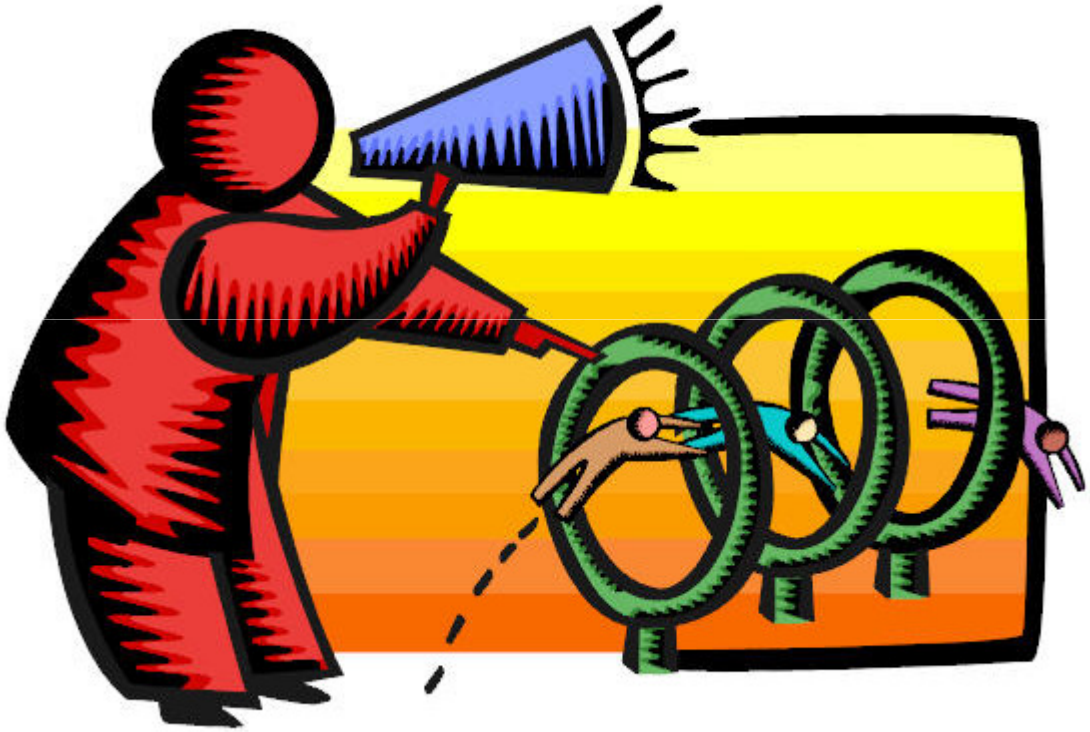
Performance tests - Data intensive

Unit	Line	Total time	Occurrences	Text	Average time	Maximum tim
ANONYMOUS BLOCK	1	0	2		0	
ANONYMOUS BLOCK	2	0	1		0	
ANONYMOUS BLOCK	5	0	1		0	
ANONYMOUS BLOCK	6	0	1		0	
ANONYMOUS BLOCK	10	0	1		0	
ANONYMOUS BLOCK	11	0	1		0	
SSP_NEZIGODE	4	0	1	procedure izracun_opasnih_mjesta	0	
SSP_NEZIGODE	11	0	1	cursor bcp_prnezg_dio_w2_cur	0	
SSP_NEZIGODE	14	0	1	select * from bcp_prnezg_dio_w2 w	0	
SSP_NEZIGODE	61	0	16	procedure insert_bcp_prnezg_opmjesta_tmp (p	0	
SSP_NEZIGODE	84	0	16	begin	0	
SSP_NEZIGODE	87	3	16	select bcp_prnezg_opmjesta_tmp_seq.nextval	0	
SSP_NEZIGODE	89	0	16	l_ce_id:= p_opasna_mjesta_coll(p_opasna_mje	0	
SSP_NEZIGODE	90	0	16	l_dio_id :=p_opasna_mjesta_coll(p_opasna_mj	0	
SSP_NEZIGODE	91	0	16	l_stac_cesta:= p_opasna_mjesta_coll(p_opasn	0	
SSP_NEZIGODE	92	0	16	l_stac_dio:= p_opasna_mjesta_coll(p_opasna_	0	
SSP_NEZIGODE	93	0	16	l_dio_id2:=p_opasna_mjesta_coll(p_opasna_m	0	
SSP_NEZIGODE	94	0	16	l_stac_cesta2:=p_opasna_mjesta_coll(p_opasn	0	
SSP_NEZIGODE	95	0	16	l_stac_dio2:=p_opasna_mjesta_coll(p_opasna_	0	
SSP_NEZIGODE	96	0	16	l_lnp_id:=p_opasna_mjesta_coll(p_opasna_mje	0	
SSP_NEZIGODE	97	0	16	l_stac_poc:= p_opasna_mjesta_coll(p_opasna_	0	
SSP_NEZIGODE	98	0	16	l_lnz_id:=p_opasna_mjesta_coll(p_opasna_mje	0	
SSP_NEZIGODE	99	0	16	l_stac_zav:= p_opasna_mjesta_coll(p_opasna_	0	
SSP_NEZIGODE	101	7	16	insert into bcp_prnezg_opmjesta_tmp(pk_ce_id	0	
SSP_NEZIGODE	122	0	115	for kk in p_opasna_mjesta_coll.first..p_opasna_	0	
SSP_NEZIGODE	124	11.046	99	insert into BCP_PRNEZG_OPMJESTA_NEZG_	112	19
SSP_NEZIGODE	126	91	99	update BCP_PRNEZG_OPMJESTA_NEZG_Th	1	

Performance tests - Data intensive

- Java - client side code
  - ====> Duration: 296 Milliseconds
- Java server side - interpreted
  - ====> Duration: 77 Milliseconds (Average on 3 tests)
- Java server side - native
  - ====> Duration: 98 Milliseconds
- PL/SQL
  - ====> Duration: 40 Milliseconds

Demo



Optimizacija SQL-a na Oracle Support način